



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/005,728	11/06/2001	Mohammad A. Abdallah	42390P5943C	2359

7590 11/16/2005
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
Seventh Floor
12400 Wilshire Boulevard
Los Angeles, CA 90025-1026

EXAMINER

ELLIS, RICHARD L

ART UNIT PAPER NUMBER

2183

DATE MAILED: 11/16/2005

Please find below and/or attached an Office communication concerning this application or proceeding.



UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
P.O. Box 1450
ALEXANDRIA, VA 22313-1450
www.uspto.gov

**BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES**

Application Number: 10/005,728
Filing Date: November 6, 2001
Effective Filing Date: March 31, 1998
Appellant(s): Abdallah et al.

MAILED

NOV 16 2005

Technology Center 2100

Lawrence M. Mennemeier, Reg. No. 51,003
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed August 10, 2005 appealing from the Office action mailed January 10, 2005.

(1) Real Party In Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is incorrect. A correct statement of the status of the claims is as follows:

Claims 1-15, 19-20, 25, 32, and 38 are canceled.

Claims 17, 26-31, and 33-37 stand rejected under 35 USC § 112, second paragraph, as indefinite.

Claims 16-18, 26-31, 35-37, and 39-42 stand rejected under 35 USC § 103(a) as unpatentable over US Patent 5,859,789 (Sidwell) in view of Visual Instruction Set (VIS TM) User's Guide, Sun Microsystems, March 1997 (Sun).

Claims 21-22, 23-24, 33-34, and 43-44 stand rejected under 35 USC § 103(a) as being unpatentable over US Patent 5,859,789 (Sidwell) in view of visual Instruction Set (VIS TM) User's Guide, Sun Microsystems, March 1997 (Sun) and further in view of US Patent 5,721,697 (Lee).

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is

incorrect. A correct status of amendments after final follows:

An amendment after final was filed on 6/9/2004, it was **not entered**.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is substantially correct. The changes are as follows:

Replace "26-38" with --26-31 and 35-37-- in line A;

Replace "16-20, 25, 26-38, and 39-42" with --16-18, 26-31, 33-37, and 39-42-- in line B.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

The following is a listing of the evidence (e.g., patents, publications, Official Notice, and admitted prior art) relied upon in the rejection of claims under appeal.

Sidwell	US Patent 5,859,789	January 12, 1999
Sun	Visual Instruction Set (VIS TM) User's Guide, Sun Microsystems	March 1997
Lee	US Patent 5,721,697	February 24, 1998
Intel	Intel Pentium TM Processor Family Developer's Manual, Volume 3: Architecture and Programming Manual	1995

TESS	Printout from Trademark Electronic Search System showing PENTIUM as a registered trademark of the Intel Corporation	May 3, 1994 registration date
Lee(2)	Subword Parallelism with MAX-2	August 1996

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

35 USC § 112, second paragraph rejection of claims 17, 26-31, and 33-37

Claim 17 reads as follows:

"The processor of Claim 16, wherein the decode unit further decodes a plurality of instructions of a PENTIUM microprocessor instruction set."

Claim 26 begins:

"A processor to execute instructions of the PENTIUM microprocessor instruction set ..."

Claims 27-31 and 33-37 inherit the indefiniteness of their parent claim, claim 26.

Claims 17, 26-31, and 33-37 are rejected under 35 USC § 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention.

Claim 17, and parent claim 26, both utilize the trademark "PENTIUM" TM as a claim limitation, which renders the claim indefinite as detailed in MPEP § 2173.05(u). MPEP § 2173.05(u) states in pertinent part:

"It is important to recognize that a trademark or trade name is used to identify a source of goods, and not the goods themselves. Thus a trademark or trade name does not identify or describe the goods associated with the trademark or trade name." (emphasis added)

"If a trademark or trade name is used in a claim as a limitation to identify or describe a particular material or product, the claim does not comply with the requirements of the [sic] 35 U.S.C. 112, second paragraph. *Ex parte Simpson*, 218 USPQ 1020 (Bd. App. 1982). The claim scope is uncertain since the trademark or trade name cannot be used properly to identify any particular material or product."

The TESS document, first entered into the record in the office action mailed January 10, 2005, clearly shows that the term PENTIUM TM is a registered trademark of the Intel Corporation. The text of

claims 17 and claim 26 both utilize the term PENTIUM™ as a claim limitation, in that the term is utilized to limit the type of instructions decoded by the decode unit (claim 17) or executed by the processor (claim 26). This is a clear attempted use of a trademark as a claim limitation, and as shown from *Ex parte Simpson* does not comply with the requirements of 35 USC 112, second paragraph. The claim scope is uncertain because the trademark PENTIUM™ cannot be properly used to identify any particular material or product.

Appellant in his arguments makes this statement:

"Such is not the case with the present claims 17 and 26 where the trademark, "PENTIUM," is being used as an adjective descriptive of the source of a well known, publicly disclosed, microprocessor instruction set."

Appellant's statement above in the arguments captures exactly why appellant's usage of the trademark PENTIUM™ in the claims renders the claims indefinite. Appellant admits that the trademark is used in the claim to represent a source of goods. However, because that trademark is also being used to limit the scope of the instructions decoded (claim 17) or executed (claim 26) by the system, it's usage does not comply with the definiteness requirements of 35 USC § 112, second paragraph. As stated in MPEP 2173.05(u) "The claim scope is uncertain since the trademark or trade name cannot be used properly to identify any particular material or product."

Appellant further references a series of nine documents submitted on October 11, 2004 as showing that the trademark PENTIUM™ has a meaning representative of one particular material or product. However, applicant's submissions do not support his assertion that the trademark is representative of any one particular material or product. In fact, applicant's submissions support a case that the trademark term has become generic and is no longer suitable for trademark protection, but not that it is representative of a particular material or product.

Appellant further refers to MPEP § 608.01(v) as supporting his assertion that the use of the trademark in the claim is permissible. However, the pertinent portion of MPEP § 608.01(v) which is significant for this appeal is as follows:

"The relationship between a trademark and the product it identifies is sometimes indefinite, uncertain, and arbitrary. The formula or characteristics of the product may change from time to time and yet it may continue to be sold under the same trademark. In patent specifications, every element or ingredient of the product should be set forth in positive, exact, intelligible language, so that there will be no uncertainty as to what is meant. Arbitrary trademarks which are liable to mean different things at the pleasure of manufacturers do not constitute such language. *Ex Parte Kattwinkle* 12 USPQ 11 (Bd. App. 1931).

In this appeal, there is great "uncertainty" as to "what is meant" because the trademark PENTIUMTM has been utilized "at the pleasure of [the] manufacturer" (Intel Corporation) to "mean different things". There are at least ten different processors produced by the Intel corporation that carry the trademark PENTIUMTM but that also contain slightly or markedly different instruction sets. E.g., a PENTIUMTM processor and a PENTIUMTM MMXTM processor are two different processors that contain two different instruction sets (the PENTIUMTM MMXTM processor contains 57 new instructions that do not exist on the PENTIUMTM processor).

Further evidence of appellant's usage of the trademark PENTIUMTM as a claim limitation can be found on pgs. 18-25 of the brief where appellant attempts to differentiate claims 17 and 26-29 from the cited references by referring to particular features of a PENTIUMTM instruction set. See for example pg. 20 where appellant states:

"Therefore, one difference between the claimed decoder of instructions of the PENTIUM microprocessor [sic] instruction set and the expected properties of the combined system of Sidwell and Sun is the absence of an expected third operand."

In this example, appellant is attempting to rely upon the trademark PENTIUMTM in the claim to incorporate a particular limitation into the claim language, namely the presence/absence of a third operand in the instruction set. However, because the trademark PENTIUMTM is not representative of any particular instruction set, can not be properly used to denote any particular instruction set or it's features, and can be changed at the whim of the trademark owner to mean something completely different yet still

be referred to as PENTIUM™, this is a clear indication that appellant is attempting to improperly utilize the trademark PENTIUM™ in the claims as a claim limitation.

Claims 16-18, 26-31, 35-37 and 39-42 are rejected under 35 USC § 103 as being unpatentable over Sidwell, US Patent 5,859,789, in view of *Visual Instruction Set (VIS™) User's Guide*, Sun Microsystems, March 1997 ("Sun"). Lee(2) (August 1996) is cited as extrinsic evidence of the general level of knowledge of one of skill in the art at the time of invention.

Sidwell taught (e.g. see figs. 1-6) the invention substantially as claimed (as per claim 16), including a data processing ("DP") system comprising:

16. A processor comprising:	fig. 1
a decode unit	fig. 1, 16, col. 3 lines 54-55, 61-64
to decode a plurality of packed data instructions	col. 5 lines 15-35, col. 6, lines 5-48, col. 7 lines 20-54
including a packed sum of absolute differences (PSAD) instruction	Sidwell did not teach that the system additionally included an instruction for performing a packed sum of absolute differences (PSAD). <u>Sun</u> taught an instruction for performing a packed sum of absolute differences (PSAD) (pg. 87-88, section 4.7.11, instruction "vis_pdist()").
having a first format to identify a first set of packed data,	As seen from col. 4 line 48 to col. 5 line 14, col. 6 lines 5-22 and 37-44, and col. 7 lines 22-27, each different instruction has a different format ("add2p" vs "mul2ps") and so when adding Sun's PSAD instruction into Sidwell's system it must inherently be given a separate format to "identify" it apart from the other existing instructions. Col. 4 lines 26-29 show that each instruction identifies from one to three source operands.
and a packed multiply-add (PMAD) instruction	col. 7 lines 20-48
having a second format	col. 7 lines 24-25, "muladd2ps"
to identify a second set of packed data,	col. 7 lines 26-30, "multiply together respective <u>pairs of objects from two operands</u> "

said decode unit to initiate a first set of operations on the first set of packed data responsive to decoding the PSAD instruction	fig. 1, "PACK OPS" signal directed to element 6, the "PACKED ARITHMETIC" unit, on the "OP" input. Col. 3 lines 61-67. When adding Sun's PSAD instruction, the decoding system would inherently also be responsible for controlling the PSAD instruction execution in the same manner as it is responsible for controlling execution of all of the other instructions that Sidwell discloses.
and to initiate a second set of operations on the second set of packed data responsive to decoding the PMAD instruction;	fig. 1, "PACK OPS" signal directed to element 6, the "PACKED ARITHMETIC" unit, on the "OP" input. Col. 3 lines 61-67
and an execution unit to perform a first operation of the first set of operations initiated by the decode unit and to perform a second operation of the second set of operations initiated by the decode unit.	fig. 1, element 6, the "PACKED ARITHMETIC" unit, which contains all of the elements diagrammed at fig. 2, col. 5 line 15 et seq, which this unit (fig. 1, element 6) is responsible for performing all the packed arithmetic operations that the system is capable of performing, which would have included both the first and second set of operations.

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to have combined Sun's packed sum of absolute differences (PSAD) instruction into Sidwell's system because Sidwell taught that the packed arithmetic unit was designed to perform additional operations (col. 5 lines 15-22) and Sun taught that a packed sum of absolute differences (PSAD) instruction was beneficial in accelerating motion compensation to support real-time video compression (pg. 88). Lee(2) shows that it was known within the general level of knowledge of one of skill in the art at the time of invention that packed data operations such as that disclosed by Sidwell, were known to be useful for and known to be utilized for performing media processing, such as video processing, which is a stated purpose for Sun's disclosed instruction. See Lee(2), pg. 51, col. 2, lines 23-47:

"One key advantage of subword parallelism is that it allows general-purpose processors to exploit wider word sizes even when not processing high-precision data. The processor can achieve more subword parallelism on lower precision data rather than wasting much of the word-oriented data paths and registers. ... A more efficient use of memory also results, since a single load- or store-word instruction moves multiple packed subwords between memory and processor registers. Hence, subword parallelism is an efficient organization for media processing, whether on a general-purpose micro-processor or a specially designed media processor or DSP."

Where Lee(2) defines "subword parallelism" as:

"A subword is a lower precision unit of data contained within a word. In subword parallelism, we pack

multiple subwords into a word and then process whole words" With the appropriate subword boundaries, this technique results in parallel processing of subwords. (Lee(2), pg. 57, col. 1, lines 30-36)

and defines "media processing" as:

"We define media processing as the processing of digital multimedia information, such as images, video, audio, 2D and 3D graphics, animation, and text." (Lee(2), pg. 57, col. 1, lines 4-7)

Because it was known by one of skill in the art that subword parallelism as detailed by Sidwell was useful for video processing, one of skill in the art would have further been motivated to include Sun's disclosed vis_pdist() instruction into a system such as Sidwell's to further enhance it's performance as a media processor, such as for processing digital video data.

As to claim 17, the further limitation provided by claim 17 is an improper attempt to limit the claim by limiting the type of instructions interpreted by the decoder to those that form a part of the PENTIUM™ microprocessor instruction set. This claim is indefinite for attempting to utilize a trademark (which only denotes a source of goods, not any particular good) as detailed in the rejection under 35 USC § 112, second paragraph (see rejection beginning on pg. 4, above). Therefore, because proper use of the trademark PENTIUM™ requires that it impart no limitation upon the claims, the claim language can not be seen as providing any limitation of any particular instruction set, and the applied references fall within the boundaries set forth by the claims.

Additionally, assuming that the trademark PENTIUM™ could impart a limitation into the claims, which it cannot, Sidwell in view of Sun did not teach that the combined system would interpret the PENTIUM™ microprocessor instruction set. However, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to have made the combined system compatible with the PENTIUM™ microprocessor instruction set because the PENTIUM™ instruction set was by far the most widely used microprocessor instruction set in the world at the time of invention, and by making the system compatible with the PENTIUM™ instruction set, the system would have been immediately

compatible and useful to the widest variety of available programs, yielding instant sales and profitability.

As to claim 18, Sun taught:

18. The processor of claim 16, wherein the first set of operations comprises:	see rejection above
a packed subtract and write carry (PSBWC) operation	pg. 87, "The pixels are subtracted from one another, pair wise". Subtraction operations inherently generate carries (known as borrows for a subtract) which is known from elementary mathematics.
a packed absolute value and read carry (PABSRC) operation;	pg. 87, "the absolute values of the differences are [taken]". Absolute value is defined from basic mathematics as $ a \equiv \begin{cases} a, & \text{if } a \geq 0 \\ -a, & \text{if } a < 0 \end{cases}$ accordingly, the carry (borrow) value must be used to determine if the result is greater than or equal to zero, or less than zero, in order to create the absolute value of the result.
and a packed add horizontal (PADDH) operation.	pg. 87, "the absolute values of the differences are accumulated into <i>accum</i> ." To accumulate the absolute values requires that all results from all absolute values be summed to a single number, i.e., a "horizontal" add if the group of packed data is viewed as lying on a horizontal axis.

As to claim 26, Sidwell in view of Sun taught the elements of the claim that are in common with claims 16-17 as detailed above in the rejections of claims 16-17. Additionally, Sidwell taught a bus to provide the first set of packed data to the execution logic for performing the first operation (fig. 1, source buses 52 and 54).

As to claim 27, Sidwell did not specifically teach that the decode-logic comprised a look-up table. However, utilizing lookup tables for decoding instruction opcodes into their basic operations has been a well known practice within the processor art for a very lengthy time (this technique is known as

microcoding) and official notice of such is hereby taken. It would have been obvious to a person of ordinary skill in the art at the time the invention was made to have microcoded Sidwell's decoder because of the well known advantages of micro-coded decoders of ease of implementation and ease of modification if a bug is discovered in the implementation. It is noted for the board that applicant has not as of the time of this appeal challenged this assertion of official notice.

As to claim 28, Sidwell's decode logic will have inherently comprised integrated circuitry.

As to claim 29, Sidwell taught that the decode logic [contained] executable operations (col. 3 lines 61-64).

As to claim 30, Sun taught:

18. The processor of claim 16, wherein the first set of operations comprises:	see rejection above
a packed subtract and write carry (PSBWC) operation	pg. 87, "The pixels are subtracted from one another, pair wise". Subtraction operations inherently generate carries (known as borrows for a subtract) which is known from elementary mathematics.
a packed absolute value and read carry (PABSRC) operation;	pg. 87, "the absolute values of the differences are [taken]". Absolute value is defined from basic mathematics as $ (a) \equiv \begin{cases} a, & \text{if } a \geq 0 \\ -a, & \text{if } a < 0 \end{cases}$ accordingly, the carry (borrow) value must be used to determine if the result is greater than or equal to zero, or less than zero, in order to create the absolute value of the result.
and a packed add horizontal (PADDH) operation.	pg. 87, "the absolute values of the differences are accumulated into <i>accum</i> ."

As to claim 31, Sidwell taught that the first format identified the first set of packed data as packed

bytes (col. 6 lines 5-20, each format ("add2p", "mul2ps", ...) identifies the first set of data as packed bytes by the presence of the "p" suffix. Sun additionally taught that the first format also identifies the first set of packed data as packed bytes (pg. 43, "vis_f32", showing a set of four packed bytes, "vis_d64", showing a set of eight packed bytes).

As to claim 35, Sidwell taught that the decode unit decoded a packed multiply-add (PMAD) instruction (col. 7 lines 23-27) having a second format ("muladd2ps" is a different format than "add2p" or "mul2ps") to identify a second set of packed data (col. 8 lines 23-24, "muladd2ps R6, R4, R5", where R6, R4, and R5 identify source and destination data registers), said decode unit to initiate a second set of operations on the second set of packed data responsive to decoding the PMAD instruction (col. 3 lines 61-64).

As to claim 36, Sidwell taught execution [sic] unit (fig. 1 element 6) to perform a second operation of the second set of operations initiated by the decode unit (col. 3 lines 61-64 and col. 7 line 20-55, each instruction being a "different operation).

As to claim 37, Sidwell taught that the second format identified the second set of packed data as packed words (col. 7 lines 23-27, the description indicates that the instruction processes 16-bit size values, and a "word" is defined as a 16-bit size value).

As to claim 39, Sidwell in view of Sun taught:

39. A processor comprising:	Sidwell: fig. 1
decode logic	Sidwell: fig. 1, 16, col. 3 lines 54-55, 61-64

to decode a packed sum of absolute differences (PSAD) instruction	Sidwell did not teach that the system additionally included an instruction for performing a packed sum of absolute differences (PSAD). <u>Sun</u> taught an instruction for performing a packed sum of absolute differences (PSAD) (pg. 87-88, section 4.7.11, instruction "vis_pdist(")).
having a first format	the format of the instruction is "vis_pdist(vis_d64 pixels1, vis_d64 pixels2, vis_d64 accumulator)"
to identify a first set of packed data,	"vis_d64 pixels1" identifies a first set of packed data
said decode logic to initiate a first set of operations on the first set of packed data responsive to decoding the PSAD instruction,	fig. 1, "PACK OPS" signal directed to element 6, the "PACKED ARITHMETIC" unit, on the "OP" input. Col. 3 lines 61-67. When adding Sun's PSAD instruction, the decoding system would inherently also be responsible for controlling the PSAD instruction execution in the same manner as it is responsible for controlling execution of all of the other instructions that Sidwell discloses.
the first set of operations comprising a packed subtract and write carry (PSUBWC) operation;	pg. 87, "The pixels are subtracted from one another, pair wise". Subtraction operations inherently generate carries (known as borrows for a subtract) which is known from elementary mathematics.
a packed absolute value and read carry (PABSRC) operation; and	pg. 87, "the absolute values of the differences are [taken]". Absolute value is defined from basic mathematics as $ (a) \equiv \begin{cases} a, & \text{if } a \geq 0 \\ -a, & \text{if } a < 0 \end{cases}$ accordingly, the carry (borrow) value must be used to determine if the result is greater than or equal to zero, or less than zero, in order to create the absolute value of the result.
a packed add horizontal (PADDH) operation;	pg. 87, "the absolute values of the differences are accumulated into <i>accum</i> ." To accumulate the absolute values requires that all results from all absolute values be summed to a single number, i.e., a "horizontal" add if the group of packed data is viewed as lying on a horizontal axis.
and execution logic to perform the first set of operations initiated by the decode logic.	fig. 1, element 6, the "PACKED ARITHMETIC" unit, which contains all of the elements diagrammed at fig. 2, col. 5 line 15 et seq, which this unit (fig. 1, element 6) is responsible for performing all the packed arithmetic operations that the system is capable of performing.

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to have combined Sun's packed sum of absolute differences (PSAD) instruction into Sidwell's system because Sidwell taught that the packed arithmetic unit performed additional operations (col. 5 lines

15-22) and Sun taught that a packed sum of absolute differences (PSAD) instruction was beneficial in accelerating motion compensation to support real-time video compression (pg. 88). As detailed above starting on pg. 8, it was within the general knowledge of one of skill in the art that Sidwell's disclosed packed data operations were utilized for accelerating media operations, such as digital video processing.

As to claim 40, this claim differs from claim 31 only by dependency and therefore is rejected for the same reasons as the rejection of claim 31 presented above.

As to claim 41, Sidwell in view of Sun taught

The processor of claim 39, wherein performing the PSUBWC operation causes the execution logic to:	see rejection of claim 39
subtract one of a plurality of elements of a first packed data of the first set of packed data from a corresponding one of a plurality of elements of a second packed data of the first set of packed data to produce a first result having a plurality of difference elements and a plurality of sign indicators	Sun: pg. 87, "The pixels are subtracted from one another, pair wise ...". "Pair wise" means to take pairs of elements (i.e., one of a first and a corresponding one of a second) and then subtract those elements. The definition of subtraction from basic mathematics indicates that a difference element (a value) and a sign are produced by the subtraction operation.
store the plurality of difference elements and the plurality of sign indicators	In order for the second operation disclosed by Sun (the absolute value) to operate, the results of the subtractions from the first operation must be stored, however temporarily, so that the absolute value can be obtained.

As to claim 42, Sidwell in view of Sun taught:

42. The processor of Claim 39, wherein performing the PABSRC operation causes the execution logic to:	see rejection of claim 39
---	---------------------------

receive a plurality of difference elements and a plurality of sign indicators;	As detailed in the rejection of claim 41, the results of the first step of Sun's instruction must be stored, however temporarily, in order to perform the absolute values of the following step. Accordingly, when performing the next step, the stored values would be "received".
produce a result data having a plurality of absolute value elements,	Sun: pg. 87, "the absolute values of the differences are [taken]".
each absolute value element produced by	The definition of absolute value from basic mathematics is $ (a) \equiv \begin{cases} a, & \text{if } a \geq 0 \\ -a, & \text{if } a < 0 \end{cases}$ As shown below, applicant has merely claimed, in very verbose fashion, the exact definition of the absolute value function from basic mathematics.
(a) subtracting one of the plurality of difference elements from a corresponding constant value if the sign indicator corresponding to that element is in a first state,	In the second line of the definition of absolute value above, $-a, \text{ if } a < 0$ the term $-a$ is equivalent to $0 - a$, or subtraction of a from zero, which is "subtracting one of the ... difference elements from a corresponding constant value (zero) if the sign indicator ... is in a first state (negative)".
or, (b) adding one of the plurality of difference elements to a corresponding constant value if the sign indicator corresponding to that element is in a second state.	In the first line of the definition of absolute value above, $a, \text{ if } a \geq 0$ the term a is equivalent to $0 + a$, or addition of a to zero, which is "adding one of [the] difference elements to a corresponding constant value (zero) if the sign indicator ... is in a second state (positive)".

Claims 21-24, 33-34, and 43-44 are rejected under 35 USC § 103 as being unpatentable over Sidwell, US Patent 5,859,789, in view of *Visual Instruction Set (VIS™) User's Guide*, Sun Microsystems, March 1997 ("Sun"), and further in view of Lee, US Patent 5,721,697. Lee(2) is cited as extrinsic evidence of the general level of knowledge of one of skill in the art at the time of invention.

Each of claims 21, 33, and 43 differ only by dependency, and each of claims 22, 34, and 44 also differ only by dependency. Accordingly, in the following, claim 21 will be taken as exemplary of claims 21, 33, and 43, and claim 22 will be taken as exemplary of claims 22, 34, and 44.

As to claim 21 (and claims 33 and 43), Sidwell in view of Sun taught the features of the claims as detailed in the rejections above. Sidwell in view of Sun did not teach the claimed aspects of claims 21, 33 or 43. However Lee taught the features of the claims as follows:

21. (and claims 33 and 43) The processor of claim 16,	see rejection of claim 16
wherein performing the first operation causes the execution unit to: produce a first plurality of partial products in a multiplier having a plurality of partial product selectors;	col. 2 lines 21-26
insert an element of a first plurality of elements of a first packed data into and substituting for bit positions of one or more of the first plurality of partial products by using partial product selectors according to the bit positions;	col. 5 lines 35-56 and table 5, note that element "abcd" is inserted into the first row partial product by product selector "1" at the top of the right hand column. Note that element "efgh" is further inserted into the fifth row partial product by product selector "1" in the fifth position of the right hand column.
and add the first plurality of elements together to produce a first result including a field comprising a sum of the first plurality of elements,	table 5, "ZZZZZ" is the result of the addition, see col. 5 lines 54-55
said field having a least significant bit.	The first Z on the right of the element ZZZZZ is the least significant bit of the field. Additionally, it should be noted that all numbers contain a least and a most significant digit.

It would have been obvious to a person of ordinary skill in the art at the time the invention was made to have combined Lee's teachings into a system containing the teachings of Sidwell and Sun because of Lee's teachings that the conventional manner of performing a tree addition is to perform plural operations over many cycles which consequently take a long time (Lee: col. 1 lines 29-35) to complete. Lee provides a system for performing a tree add operation in fewer cycles with only minor changes to an existing multiplication circuit (Lee: col. 2 lines 1-2) for the purposes of accelerating video compression operations (Lee: col. 1 lines 20-29 and 36-40), which is a purpose of Sun (Sun: pg. 88, "The vis_pdist() instruction is intended to accelerate motion compensation to support real-time video compression ...". As

detailed above starting on pg. 8, it was within the general knowledge of one of skill in the art that Sidwell's disclosed packed data operations were utilized for accelerating media operations, such as digital video processing.

As to claim 22 (and claims 34 and 44), Lee taught shifting the first result to produce a second result having a least significant bit position and to align the least significant bit of the field with the least significant bit position of the second result (col. 5 lines 55-56).

As to claim 23, Sidwell in view of Sun and Lee taught:

23 A processor comprising:	Sidwell: fig. 1
a decode unit	Sidwell: fig. 1, 16, col. 3 lines 54-55, 61-64
to decode a plurality of packed data instructions	Sidwell: col. 5 lines 15-35, col. 6, lines 5-48, col. 7 lines 20-54
including a packed sum of absolute differences (PSAD) instruction	Sidwell did not teach that the system additionally included an instruction for performing a packed sum of absolute differences (PSAD). <u>Sun</u> taught an instruction for performing a packed sum of absolute differences (PSAD) (pg. 87-88, section 4.7.11, instruction "vis_pdist()").
having a first format to identify a first set of packed data,	Sun: As seen from col. 4 line 48 to col. 5 line 14, col. 6 lines 5-22 and 37-44, and col. 7 lines 22-27, each different instruction has a different format ("add2p" vs "mul2ps") and so when adding Sun's PSAD instruction into Sidwell's system it must inherently be given a separate format to "identify" it apart from the other existing instructions. Col. 4 lines 26-29 show that each instruction identifies from one to three source operands.
and a packed multiply-add (PMAD) instruction	Sidwell: col. 7 lines 20-48
having a second format	Sidwell: col. 7 lines 24-25, "muladd2ps"
to identify a second set of packed data,	Sidwell: col. 7 lines 26-30, "multiply together respective <u>pairs of objects from two operands</u> "

said decode unit to initiate a first set of operations on the first set of packed data responsive to decoding the PSAD instruction	Sidwell: fig. 1, "PACK OPS" signal directed to element 6, the "PACKED ARITHMETIC" unit, on the "OP" input. Col. 3 lines 61-67. When adding Sun's PSAD instruction, the decoding system would inherently also be responsible for controlling the PSAD instruction execution in the same manner as it is responsible for controlling execution of all of the other instructions that Sidwell discloses.
and to initiate a second set of operations on the second set of packed data responsive to decoding the PMAD instruction;	Sidwell: fig. 1, "PACK OPS" signal directed to element 6, the "PACKED ARITHMETIC" unit, on the "OP" input. Col. 3 lines 61-67
and an execution unit to perform a first operation of the first set of operations initiated by the decode unit and to perform a second operation of the second set of operations initiated by the decode unit;	Sidwell: fig. 1, element 6, the "PACKED ARITHMETIC" unit, which contains all of the elements diagrammed at fig. 2, col. 5 line 15 et seq, which this unit (fig. 1, element 6) is responsible for performing all the packed arithmetic operations that the system is capable of performing, which would have included both the first and second set of operations.
wherein performing the first operation causes the execution unit to: produce a first plurality of partial products in a multiplier having a plurality of partial produce selectors,	Lee: col. 2 lines 21-26
insert an element of a first plurality of elements of a first packed data into and substituting for bit positions of one or more of the first plurality of partial products by using partial product selectors corresponding to the bit positions,	Lee: col. 5 lines 35-56 and table 5, note that element "abcd" is inserted into the first row partial product by product selector "1" at the top of the right hand column. Note that element "efgh" is further inserted into the fifth row partial product by product selector "1" in the fifth position of the right hand column.
and add the first plurality of elements together to produce a first result including a field comprising a sum of the first plurality of elements,	Lee: table 5, "ZZZZZ" is the result of the addition, see col. 5 lines 54-55
said field having a least significant bit;	Lee: The first Z on the right of the element ZZZZZ is the least significant bit of the field. Additionally, it should be noted that all numbers contain a least and a most significant digit.

and wherein performing the second operation causes the execution unit to: produce a second plurality of partial products in the multiplier having the plurality of partial product selectors,	Lee: col. 2 lines 21-26
the second plurality of partial products comprising four distinct sets of partial products	Lee: col. 6, table 6, there are four distinct sets ("abcd", "efgh", "ijkl", and "mnop" shown in the partial products.
including a first set of partial products corresponding to a first product for elements of the second set of packed data,	Sidwell: fig. 6, 114, showing multiplication of S2[3] with S1[3], which when Lee's teachings are introduced into Sidwell and Sun's system, would provide the entry for Lee's col. 6, table 6, "abcd"
a second set of partial products corresponding to a second product for elements of the second set of packed data,	Sidwell: fig. 6, 112, showing multiplication of S2[2] with S1[2], which when Lee's teachings are introduced into Sidwell and Sun's system, would provide the entry for Lee's col. 6, table 6, "efgh"
a third set of partial products corresponding to a third product for elements of the second set of packed data,	Sidwell: fig. 6, 110, showing multiplication of S2[1] with S1[1], which when Lee's teachings are introduced into Sidwell and Sun's system, would provide the entry for Lee's col. 6, table 6, "ijkl" element.
and a fourth set of partial products corresponding to a fourth product for elements of the second set of packed data,	Sidwell: fig. 6, 108, showing multiplication of S2[0] with S1[0], which when Lee's teachings are introduced into Sidwell and Sun's system, would provide the entry for Lee's col. 6, table 6, "mnop" element.
and add the first set of partial products together with the second set of partial products to produce a first distinct element of a packed result	Sidwell: 118, showing summing together the partial products output from multipliers 114 and 112, this corresponds to elements "abcd" and "efgh" of Lee's col. 6, table 6.
and add the third set of partial products together with the fourth set of partial products to produce a second distinct element of the packed result.	Sidwell: 116, showing summing together the partial products output from multipliers 110 and 108, this corresponds to elements "ijkl" and "mnop" of Lee's col. 6, table 6.

(10) Response to Argument

Response to applicant's argument section entitled "1. Claims 17 and 26-29 Are Not Obvious"

At pg. 18-19 of the brief, appellant summarizes what he interprets Sidwell and Sun to disclose. At

pg. 20 of the brief, appellant makes the following argument regarding the claim language (as per claims 17 and 26-29) in view of the references:

"Appellant respectfully submits that since the PENTIUM microprocessor instruction set has a well known opcode format, which permits two operands, one of the operands acting both as a source operand and a destination operand. To decode and execute the `vis_pdist()` instruction of Sun having three source operands in a processor for executing two-operand instructions of the PENTIUM microprocessor instruction set is not suggested by either of the cited references."

Appellant's argument is not persuasive for several reasons. First, because appellant is clearly attempting to use a trademark (PENTIUM™) as a direct claim limitation, the claim is indefinite under 35 USC § 112, second paragraph. Accordingly, because of this indefiniteness, the claim can not be seen as requiring any particular number of operands for any instruction because the trademark PENTIUM™ only identifies a source of an instruction set, and not any particular single instruction set. Accordingly, because of the presence and improper use of the trademark PENTIUM™ in the claims, the claims can not be interpreted as containing any limitation upon the number of operands the claimed instruction utilizes. As such, the claim is broad enough that it reads upon any number of operands, and therefore, references showing three operand instructions fall within the boundaries set forth by the claim language.

Second, the claim language itself, wholly apart from the trademark PENTIUM™ provides no limitation of any particular number of operands required by the instructions. The claim language is simply broad, and as such, reads upon any number of operands in the instruction. Appellant is reminded that it is the claim language which defines the invention, not the specification nor other material:

"The invention disclosed in Hiniker's written description may be outstanding in its field, but the name of the game is the claim." *In re Hiniker Co.*, 47 USPQ2d 1523, 1529 (Fed. Cir. 1998)

Third, as will be seen below starting at pg. 22, Sun's system naturally suggests a two operand format for their instruction because one of the three operands is required to also be a destination operand referencing the same register.

At pg. 21 of the brief, appellant argues:

"Yet, because the instructions of the PENTIUM microprocessor [sic] instruction set require only two sources, one of which is also a destination, the data path for the packed sum of absolute differences may be 75% as wide as one requiring three sources and the number of read ports required in the register file may be 66% as many as would otherwise be required for reading three source registers. Such reductions are statistically significant."

This is not found persuasive because appellant is reminded that the claim language contains absolutely no language indicating any limitations upon a width of a data path or number of register read ports in order for a reference to be within the boundaries of the claim language. Appellant is now directly arguing completely unclaimed features, which is improper.

Claimed subject matter, not the specification, is the measure of invention. Limitations in the specification cannot be read into the claims for the purpose of avoiding the prior art. *In re Self*, 213 USPQ 1,5 (CCPA 1982); *In re Priest*, 199 USPQ 11,15 (CCPA 1978).

"It is the claims that measure the invention." *SRI Int'l v. Matsushita Elec. Corp.*, 775 F.2d 1107, 1121, 227 USPQ 577, 585 (Fed. Cir. 1985) (en banc).

"The invention disclosed in Hiniker's written description may be outstanding in its field, but the name of the game is the claim." *In re Hiniker Co.*, 47 USPQ2d 1523, 1529 (Fed. Cir. 1998).

"[A]s an initial matter, the PTO applies to the verbiage of the proposed claims the broadest reasonable meaning of the words in their ordinary usage as they would be understood by one of ordinary skill in the art, taking into account whatever enlightenment by way of definitions or otherwise that may be afforded by the written description contained in the applicant's specification." *In re Morris*, 44 USPQ2d 1023, 1027 (Fed. Cir. 1997).

"limitations appearing in the specification will not be read into the claims, and ... interpreting what is meant by a word in a claim 'is not to be confused with adding an extraneous limitation appearing in the specification, which is improper'." *Intervet Am., v. Kee-Vet Labs.*, 12 USPQ2d 1474, 1476 (Fed. Cir. 1989)(citation omitted).

"it is entirely proper to use the specification to interpret what the patentee meant by a word or phrase in the claim, ... this is not to be confused with adding an extraneous limitation appearing in the specification, which is improper. By 'extraneous,' we mean a limitation read into a claim from the specification wholly apart from any need to interpret ... particular words or phrases in the claim." *In re Paulsen*, 31 USPQ2d 1671, 1674 (Fed. Cir. 1994) (citation omitted).

At pg. 21-22 of the brief, appellant argues:

"Appellant respectfully notes that the three IMUL instructions listed that use implicit operands use only one programmer specified operand and an implicit destination register where the lower half of

the destination register is used as a source (pg. 25-165, lines 3-5). Thus, the IMUL instructions with implicit operands are effectively, still, two-operand instructions."

This argument is not persuasive because appellant is continuing to argue unclaimed features of the claim language, which as pointed out above starting at pg. 21, is improper. Appellant's arguments appear to be that because the IMUL instruction does not allow three programmer specified operands, it can not be termed a three operand instruction. But not only does appellant's claim language make no mention of any number of operands, the claim language also makes no reference to any number of programmer specified operands vs. implicit operands. The cited IMUL instruction obtains three source pieces of data in order to perform it's operation. As seen from pg. 25-165, in the first table on the page, the sixth IMUL instruction (IMUL r16,r/m 16, imm8) contains three input operands ("r16" a register to read, "r/m 16" a register or memory address to read, and "imm8" an immediate value). Therefore, this is a three operand instruction, and is evidence that in this one particular implementation of an instruction set carrying the trademark PENTIUM™, three operand instructions were indeed implemented.

At pg. 22 of the brief, appellant argues (emphasis added):

"Therefore, appellant respectfully submits that since the PENTIUM microprocessor [sic] instruction set has an opcode format that permits two operands, one of those operands acting both as a source operand and a destination operand, and since the vis_pdist() instruction of Sun requires three source operands, one of which is also a destination (p. 87, 4.7.11 Syntax and Description); a sum of absolute differences instruction with a two-operand opcode format would not perform the operation defined by the vis_pdist() instruction of Sun without modification."

This is not found persuasive because as highlighted above, appellant is playing the classic hide the ball shell game in counting the number of operands. On one hand, appellant argues that the PENTIUM™ instruction format is two operand with one operand acting as both source and destination, yet on the other hand appellant is arguing that Sun is three operand with two of the three acting as both source and destination. By appellant's logic, the PENTIUM™ instruction set is three operand because one acts as both source an destination just as much as Sun's instruction is not two operand because one acts as both

source and destination. Appellant can not count operands in two different ways for the two different instruction sets because doing so results in a false impression of a difference. Either an operand that acts as both source and destination is one operand (as appellant counts it for a PENTIUM™ system), or it is two operands (as appellant counts it for Sun's system), but it can not simultaneously be both.

In either case, appellant's arguments show that the instruction sets are more compatible than different, because since Sun requires one operand to be both a source and a destination (see Sun, pg. 87 (emphasis added), "To use vis-Pdist() from C, it is necessary for the accumulating register *accumulator* to appear both as an argument and as the receiver of the return value"), that one operand logically maps directly to appellant's argued PENTIUM™ implementation of an operand that acts as both a source and a destination.

At pg. 22 of the brief, appellant argues:

"the vis_pdist() instruction of Sun computes an accumulation of current and prior absolute differences ... There is no accumulation of prior absolute differences in what appellant has done."

This is not found persuasive because appellant is arguing that because Sun is performing an additional function beyond the minimal functions in appellant's claims, that the reference does not read upon the claims. Appellant is in complete error in this argument. Appellant is reminded that the claims are written in an open ended claim form by use of the transitional phrase "comprising". As stated in MPEP § 2111.03, the term comprising does not exclude other elements from the references:

The transitional term "comprising", which is synonymous with "including," "containing," or "characterized by," is inclusive or open-ended and does not exclude additional, unrecited elements or method steps. See, e.g., *Invitrogen Corp. v. Biocrest Mfg., L.P.*, 327 F.3d 1364, 1368, 66 USPQ2d 1631, 1634 (Fed. Cir. 2003) ("The transition comprising' in a method claim indicates that the claim is open-ended and allows for additional steps."); *Genentech, Inc. v. Chiron Corp.*, 112 F.3d 495, 501, 42 USPQ2d 1608, 1613 (Fed. Cir. 1997) ("Comprising" is a term of art used in claim language which means that the named elements are essential, but other elements may be added and still form a construct within the scope of the claim.); *Moleculon Research Corp. v. CBS, Inc.*, 793 F.2d 1261, 229 USPQ 805 (Fed. Cir. 1986); *In re Baxter*, 656 F.2d 679, 686, 210 USPQ 795, 803 (CCPA 1981); *Ex parte Davis*, 80 USPQ 448, 450 (Bd. App. 1948) ("comprising" leaves "the claim open for the inclusion of unspecified ingredients even in major amounts").

Therefore, in a first case, appellant's claim language, by use of the transitional phrase "comprising" does not exclude anything and instead "leaves 'the claim open for the inclusion of unspecified ingredients even in major amounts'". Next, appellant's claim language itself must be analyzed to determine if it in any way directly excludes an accumulation of prior absolute differences from the scope of the claim language. The claims define that there is an instruction which is "a packed sum of absolute differences (PSAD) instruction". The claim language provides no further limitation of the PSAD instruction. This language requires only that the prior art provide an instruction which performs a packed sum of absolute differences. The language is broad enough that it encompasses both instructions which perform prior accumulations, as well as instructions which do not perform prior accumulations, because both types of instructions perform "packed sums of absolute differences". Accordingly, the breadth of the claim language is sufficient that it covers Sun's particular instruction implementation.

At pgs. 22-23 of the brief, appellant argues (appellant's emphasis unchanged):

"Additionally, Sun discloses that in the `vis_pdist()` instruction, "it is necessary for the accumulator register to appear both as an argument and as the receiver of the return value" (p. 88, first paragraph, emphasis supplied). Thus, Sun teaches away from an implicit source that is also the destination register, which is precisely the technique employed in the implicit-operand IMUL instructions, of the PENTIUM microprocessor [sic] instruction set."

This is not found persuasive because as detailed above, appellant is counting Sun's use of the same register as source and destination as two operands, while counting the PENTIUM™ system's use of the same register as source and destination as one operand. Appellant can not count the same usage of registers in two different manners. Use of the same register as source and destination is either one operand, or it is two operands, but it can not simultaneously be both one and two operands. Additionally, because Sun clearly taught that the same register must be used as both a source and a destination, this is a clear suggestion by Sun of use of the PENTIUM™ system method of one register as source and destination, and therefore, the teachings of Sun clearly indicate that the disclosed instruction is indeed

compatible with the PENTIUM™ system use of one register as source and destination. That one register would correspond exactly to Sun's register that is used as source and destination.

At pg. 23 of the brief, appellant argues:

"appellant respectfully submits that the packed sum of absolute differences (PSAD) instruction of the present application does not have an implicit operand."

This is not found persuasive because at no point within any of the claims of the present application is there an express recitation that the PSAD instruction does not have an implicit operand. The claim language makes no mention of the number of operands the PSAD instruction utilizes, much less whether all of those operands are explicit or implicit. Appellant is reminded that it is the claim language that defines the invention, not some unstated requirement:

"The invention disclosed in Hiniker's written description may be outstanding in its field, but the name of the game is the claim." *In re Hiniker Co.*, 47 USPQ2d 1523, 1529 (Fed. Cir. 1998)

The present claim language is simply broad, broad enough that a PSAD instruction with all implicit operands, a PSAD instruction with 50% implicit and 50% explicit, and a PSAD instruction with all explicit operands all will read upon and fall within the boundaries set out by the claim language.

At pg. 24 of the brief, appellant argues:

"Finally, Sidwell's system provides no path for an accumulator input to packed arithmetic unit 6, either from result but 56 or as a third source operand to packed arithmetic unit 6 ... Therefore, Sidwell's system could not perform Sun's packed sum of absolute differences without significant modifications to permit a third source operand for packed arithmetic instructions."

Appellant's argument is not persuasive for two reasons. First, appellant's argument is that the Sun and Sidwell references are not physically combinable because appellant's opinion is that Sidwell did not implement a component necessary for Sun's system to operate. However, as shown in MPEP § 2143(III), such an argument is insufficient to overcome a rejection of obviousness:

"III. ARGUING THAT PRIOR ART DEVICES ARE NOT PHYSICALLY COMBINABLE

"The test for obviousness is not whether the features of a secondary reference may be bodily incorporated into the structure of the primary reference.... Rather, the test is what the combined teachings of those references would have suggested to those of ordinary skill in the art." In re Keller, 642 F.2d 413, 425, 208 USPQ 871, 881 (CCPA 1981). See also In re Sneed, 710 F.2d 1544, 1550, 218 USPQ 385, 389 (Fed. Cir. 1983) ("[I]t is not necessary that the inventions of the references be physically combinable to render obvious the invention under review."); and In re Nievelt, 482 F.2d 965, 179 USPQ 224, 226 (CCPA 1973) ("Combining the teachings of references does not involve an ability to combine their specific structures."). However, the claimed combination cannot change the principle of operation of the primary reference or render the reference inoperable for its intended purpose. See MPEP § 2143.01."

Additionally, appellant's argument is an argument against the references individually, because appellant is arguing a singular feature of the Sidwell reference without consideration of the rejection being over a combination of references. As shown in MPEP § 2143(IV) this is also improper:

"IV. ARGUING AGAINST REFERENCES INDIVIDUALLY

One cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. In re Keller, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); In re Merck & Co., Inc., 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986)."

And finally, Sidwell himself specifically makes references to the instructions of his system as having from one to three source operands (col. 4 lines 25-29). Therefore, Sidwell himself specifically indicated that three source operands were within the scope of his invention, therefore, assuming by appellant's counting method that Sun did have three operands, Sidwell did foresee three operand instructions and therefore the references are indeed combinable.

Response to appellant's argument section entitled "2. Claims 18, 30 and 39-42 Are Not Obvious"

At pg. 26-27 of the brief, appellant argues:

"Sun does not teach a packed subtract and write carry operation or a packed absolute value and read carry operation, as set forth in claims 18, 30 and 39."

...

"Sidwell states that 'The execution units 2, 4, 6 do not hold any state between instructions. Thus subsequent instructions are independent.'" (col. 4, lines 36-38)

Therefore, Sidwell teaches away from an execution unit to hold carry state between instructions, which is what appellant has done in the packed subtract and write carry operation, and the packed absolute value and read carry operation as set forth by claims 18, 30 and 39.

This is not found persuasive because appellant is overlooking the fact that his claim language states

that there exists a single instruction (PSAD) (claim 16) which results in a "set of operations" (claim 16) in which dependent claims 18, 30 and 39 further define that "set of operations" of that **one** PSAD instruction as the claimed "packed subtract ...", "packed absolute value ..." and "packed add horizontal ..." operations. In other words, appellant has structured his claim language to define that **one single** PSAD instruction performs three sub-operations. Therefore, the fact that Sidwell may or may not hold any state between instructions is immaterial, because the "instructions" Sidwell refers to corresponds to appellant's PSAD instruction, not to appellant's sub-operations. As to appellant's assertion that Sun does not teach a packed subtract and write carry or a packed absolute value and read carry, appellant's attention is directed to the rejection of claim 18 above (see pgs. 10+ above) where it is explained how Sun's description requires that the carries be both stored and read.

At pg. 27-28 of the brief, appellant argues:

"There is no accumulation of prior absolute differences in the packed sum of absolute differences of the instant claims at issue."

This argument has already been answered in depth at pgs. 23+ above and pgs. 31+ below.

At pg. 28 of the brief, appellant argues:

"Because the packed sum of absolute differences instruction does not require an accumulator source, the data path for the packed sum of absolute differences may be 75% as wide as one requiring a third source and the number of read ports required in the register file may be 66% as many as would otherwise be required for reading a third source register."

This argument has already been answered in depth at pgs. 21+ above and pgs. 32+ below.

At pg. 28 of the brief, appellant argues:

"Neither cited reference discusses or suggests the writing of any carry state as part of a packed subtraction operation or the reading of any carry state as part of a packed absolute value operation."

This argument has already been addressed in depth in the rejection of claim 18 starting on pg. 10 above.

At pgs. 28-29 of the brief, appellant argues:

"What Sidwell teaches is (col. 4, lines 36-38, emphasis supplied) that, 'The execution units 2, 4, 6 do not hold any state between instructions. Thus subsequent instructions are independent.' Therefore, appellant respectfully submits that the presence of such unexpected operations to hold carry state between subsequent instructions is evidence of nonobviousness."

This argument has been treated in depth at pgs. 26+ above.

At pg. 29 of the brief, appellant argues:

"Because the packed subtract and write carry operation and the packed absolute value and read carry operation employ an execution unit to hold carry state between subsequent instructions rather than duplicating the adder/subtractor circuitry, the execution circuitry for performing the packed subtract and write carry operation and the packed absolute value and read carry operation may be 50% of the execution circuitry for performing an absolute difference operation without an execution unit to hold carry state between instructions."

This argument is not persuasive because as detailed starting at pg. 10 and 26 above, Sun's description of the operation of the PSAD instruction requires that carry state be held. Furthermore, appellant is additionally arguing yet another unclaimed feature of the invention, namely this supposed 50% reduction in execution circuitry. Appellant's claims place no limits upon, and are therefore broad enough, that they read upon and are rendered obvious by systems which contain full circuitry and systems which contain only 50% of the circuitry. This is because appellant's claims only claim the function of performing the operations and holding the carries, but do not place any structural limitations upon the circuitry responsible for performing the operations and holding the carries. Lacking these structural limitations, appellant can not have the claims read over the prior art based upon unclaimed subject matter.

"The invention disclosed in Hiniker's written description may be outstanding in its field, but the name of the game is the claim." *In re Hiniker Co.*, 47 USPQ2d 1523, 1529 (Fed. Cir. 1998)

At pgs. 29-30 of the brief, appellant argues:

"The Final Office Action of April 9, 2004 (7.4 of paper no. 8) states that a packed subtract and write carry operation and a packed absolute value and read carry operation are inherently present in Sun's system. Appellant respectfully disagrees.

Two commonly used techniques for computing an absolute differences found in the design of floating point mantissa arithmetic are: (1) compare two numbers and reorder to subtract the smaller from the larger, or (2) subtract the two numbers in both directions and select the positive result. Appellant respectfully submits that one of these two alternatives may reasonably be expected to be inherently present in Sun's system rather than the packed subtract and write carry operation and the packed absolute value and read carry operation set forth by the present application."

This is not found persuasive because appellant has completely overlooked the clear teachings provided by the Sun reference. Appellant's attention is directed to cited pg. 87 of the Sun reference, where Sun very clearly details that, **first**, a subtraction occurs ("The pixels are subtracted from one another, pair wise") and that **second**, an absolute value is taken ("and the absolute values of the differences are accumulated"). In order to perform the "absolute values" of the "**differences**", the "**differences**" must exist. Therefore, Sun is clearly indicating that the algorithm chosen was, **first**, subtract, **second** take the absolute value of the result of the subtraction. This description by Sun of their algorithm is why it is inherent in Sun's system that the carries (properly termed "borrows") from the subtraction be maintained for use by the absolute value function. Further explanation of this maintaining of the carries can be found starting on pg. 10 above. Accordingly, the fact that appellant has found two alternate methods of performing an absolute value is not pertinent because neither of appellant's two proposed methods provide for Sun's clearly disclosed method of subtract first, take absolute value of the result of the subtraction second. Appellant's first proposed alternative method compares first and subtracts second, which fails to match Sun's disclosure because Sun discloses no comparison operation. And appellant's second proposed alternative method performs two subtractions followed by a selection, not Sun's clearly disclosed single subtraction and absolute value of the result of that single subtraction. Therefore, the mere fact that appellant has found another inventors method of performing an absolute value does not remove the clear teachings of the Sun disclosure of the exact claim language method steps of first performing a subtract

operation, and second performing an absolute value upon the results of that subtraction operation (see Sun, pg. 87, "The pixels are subtracted from one another, pair wise, and the absolute values of the differences are accumulated into *accum*).

Appellant next on pgs. 30-31 attempts to bolster his point above by reference to the Van Hook patent. However, as detailed in the preceding paragraph, appellant's alternate methods do not fit within Sun's description of the method of how their instruction operates, so the fact that Van Hook performed an absolute value differently simply means that there are multiple methods of performing absolute values. It does not change the fact that Sun's explicit disclosure requires that the carries be maintained for correct operation as explained above.

At pg. 31 of the brief, appellant argues:

"Additionally, the instant claims at issue set forth decode logic to initiate a set of operations responsive to decoding the PSAD instruction, the operations comprising: a packed subtract and write carry (PSUBWC) operation; a packed absolute value and read carry (PABSRC) operation; and a packed add horizontal (PADDH) operation. ... Decoding instructions into such sets of control signals and/or microcode entry points is not disclosed by Sidwell or by Sun."

This is not found persuasive because appellant is again confusing claimed subject matter with disclosed subject matter. It has been shown in the rejection of the claims above that Sidwell does decode instructions into control signals for the execution unit (Sidwell: fig. 1, 16, "OPCODE", "PACK OPS", 6, "OP", fig. 2, "OP", 82, "ROUTE OPCODE"). Sun clearly taught that the *vis_pdist* instruction contained exactly the three steps required by applicant's claims in the same order claimed, (Sun: pg. 87, last paragraph). However, appellant's claims require only that these three operations be performed. They contain no limiting language limiting the scope of the claims to any particular method or manner of initiating or performing these three operations. Accordingly, appellant's claim language is simply broad, broad enough that it encompasses any manner of implementing these three operations. Therefore, since

the claim language does not require "microcode entry points", appellant can not rely upon this teaching from the specification to make the claim read over the prior art of record. As Sidwell already taught "control signals" (fig. 1, "PACK OPS") the decoding to control signals aspect that is claimed is indeed taught by the references.

Appellant further on pgs. 32-33 again refers to the Van Hook reference to attempt to support his point above. However, appellant is reminded that the rejections are based upon Sidwell in view of Sun, and further that appellant's claims do not require any particular microcode entry point decoding as argued by appellant.

Response to appellant's brief section entitled "3. Claims 16 and 36 are Not Obvious"

At pg. 35 of the brief, appellant argues:

"Therefore, in the combined system of Sidwell and Sun, the vis_pdist() instruction would be expected to compute an accumulation of current and prior absolute differences rather than a sum of the absolute differences on the first identified set of packed data as set forth in the instant claims at issue. There is no accumulation of prior absolute differences in the packed sum of absolute differences of the instant claims at issue."

Appellant argues that because Sun is performing an additional function beyond the minimal functions in appellant's claims, that the reference does not read upon the claims. Appellant is in complete error in this argument. Appellant is reminded that claim 16 is written in an open ended claim form by use of the transitional phrase "comprising". As stated in MPEP § 2111.03, the term comprising does not exclude other elements from the references:

The transitional term "comprising", which is synonymous with "including," "containing," or "characterized by," is inclusive or open-ended and does not exclude additional, unrecited elements or method steps. See, e.g., *Invitrogen Corp. v. Biocrest Mfg., L.P.*, 327 F.3d 1364, 1368, 66 USPQ2d 1631, 1634 (Fed. Cir. 2003) ("The

transition comprising' in a method claim indicates that the claim is open-ended and allows for additional steps."); *Genentech, Inc. v. Chiron Corp.*, 112 F.3d 495, 501, 42 USPQ2d 1608, 1613 (Fed. Cir. 1997) ("Comprising" is a term of art used in claim language which means that the named elements are essential, but other elements may be added and still form a construct within the scope of the claim.); *Moleculon Research Corp. v. CBS, Inc.*, 793 F.2d 1261, 229 USPQ 805 (Fed. Cir. 1986); *In re Baxter*, 656 F.2d 679, 686, 210 USPQ 795, 803 (CCPA 1981); *Ex parte Davis*, 80 USPQ 448, 450 (Bd. App. 1948) ("comprising" leaves "the claim open for the inclusion of unspecified ingredients even in major amounts").

Therefore, in a first case, appellant's claim language, by use of the transitional phrase "comprising" does not exclude anything and instead "leaves 'the claim open for the inclusion of unspecified ingredients even in major amounts'". Next, appellant's claim language itself must be analyzed to determine if it in any way directly excludes an accumulation of prior absolute differences from the scope of the claim language. Claim 16 defines that there is an instruction which is "a packed sum of absolute differences (PSAD) instruction". The claim language provides no further limitation of the PSAD instruction. This language requires only that the prior art provide an instruction which performs a packed sum of absolute differences. The language is broad enough that it encompasses both instructions which perform prior accumulations, as well as instructions which do not perform prior accumulations, because both types of instructions perform packed sums of absolute differences. Accordingly, the breadth of the claim language is sufficient that it covers Sun's particular instruction implementation.

Appellant next expends four additional pages of the brief (pgs. 36-39) referring to the Van Hook reference in a further attempt to argue that a PSAD instruction that accumulated prior results would not read upon his claim language. None of this discussion by appellant changes the above analysis showing that the claim provides no exclusionary language excluding such an instruction from its scope.

At pgs. 39-40 of the brief, appellant argues:

"Because the packed sum of absolute differences instruction does not require an accumulator source, the data path for the packed sum of absolute differences (requiring only two sources and one destination) may be 75% as wide as one also requiring a third source, which is statistically significant."

In response to this argument it is sufficient to point out to appellant that the language of claim 16

provides no reference at all to any number of sources, and as such, one source, two source, three source, or infinite source instructions would all fall within the boundaries outlined by the claim language.

At pg. 40 of the brief, appellant argues:

"Also since there is no version of the vis_pdist() instruction that does not use an accumulation of prior absolute differences, an additional instruction, vis_fzero(), is required by Sun to initialize the accumulator before the vis_pdist() instruction can be used (e.g. see Sun, p. 88, line 9-10)."

This argument by appellant is the same argument already answered above with respect to the comprising language of the claim. The same reasoning presented above applies to this argument, that the claim is open ended and therefore, the prior art is allowed to contain additional elements (e.g., a vis_fzero() instruction) and still render the claim unpatentable.

At pg. 40 of the brief, appellant argues:

"Claim 6 also sets forth an execution unit to perform a first operation of the first set of operations initiated by the decode unit responsive to decoding the PSAD instruction and to perform a second operation of the second set of operations initiated by the decode unit responsive to decoding the PMAD instruction. Since Sun does not disclose a multiply-add instruction and since Sidwell does not disclose a sum of absolute differences, neither of the references discloses or suggests an execution unit to perform an operation initiated by the decode unit responsive to decoding the PSAD instruction and an operation initiated by the decode unit responsive to decoding the PMAD instruction."

In this argument, appellant has merely argued against the references individually, which is improper:

IV. ARGUING AGAINST REFERENCES INDIVIDUALLY

"One cannot show nonobviousness by attacking references individually where the rejections are based on combinations of references. In re Keller, 642 F.2d 413, 208 USPQ 871 (CCPA 1981); In re Merck & Co., Inc., 800 F.2d 1091, 231 USPQ 375 (Fed. Cir. 1986)." MPEP § 2145(IV)

Appellant's argument is merely that Sidwell does not disclose the instruction of Sun and Sun does not disclose the instruction of Sidwell that neither reference discloses an execution unit performing both operations. However, the rejection is a combination, that of combining the teachings of Sun into Sidwell.

Sidwell discloses a packed arithmetic unit (element 6 of fig. 1) which performs packed arithmetic, and as well discloses that inside the packed arithmetic unit there is available space for additional functions (fig. 2, elements 72 and 78, col. 5 lines 21-22, "other packed arithmetic units 72,78"). Accordingly, one of skill in the art would have been motivated to simply insert Sun's instruction into Sidwell's system as one of the "other packed arithmetic units" 72 or 78.

On pgs. 40-41 of the brief, appellant argues:

"Sidwell admits that the multiply-add unit is capable of executing a single instruction, the result of executing that instruction being to multiply together respective pairs of objects from two operands and to add together the results to provide a final result (col. 7, lines 21-31, emphasis supplied).

Since Sidwell teaches that the multiply-add execution unit 76 could perform only the operations initiated by the decode unit responsive to decoding a packed multiply-add instruction, Sidwell teaches away from the multiply-add execution unit 76 performing a first operation initiated by the decode unit responsive to decoding the PSAD instruction." (emphasis unchanged).

Appellant's appear to have become confused as to what element from Sidwell was referenced as the claimed "an execution unit to perform a first operation ... and ... a second operation ... initiated by the decode unit". Appellant appears to be under the incorrect impression that multiply add unit 76 was referenced as the execution unit. However, appellant's attention is directed to the rejection where it is clearly indicated that the unit which would be performing the "first operation" and the "second operation" is unit 6 on fig. 1, the "PACKED ARITHMETIC" unit, i.e., the unit responsible for performing all packed arithmetic operations, be they PMAD, PSAD, or otherwise.

On pg. 41 of the brief, appellant argues:

"Moreover, without viewing the prior art in retrospect with the aid of appellant's disclosure, there is no suggestion in the cited references of an execution unit, as set forth by the instant claims at issue, to perform an operation initiated responsive to decoding the PSAD instruction and also an operation initiated responsive to decoding the PMAD instruction."

This argument by appellant is merely a reiteration of the argument answered immediately above.

Appellant is arguing that there is no execution unit in Sidwell which performs multiple different packed data operations. However, as is clearly seen from fig. 1, there is indeed an execution unit (element 6) which is responsible for the execution of at least eight different packed data instructions (seven shown on col. 6, one shown on col. 7). Therefore, Sidwell has already detailed an execution unit that performs multiple different packed data operations, and simply adding a ninth packed data operation into that unit from Sun would therefore have been obvious as previously set forth.

Appellant next expends another page (pg. 42) referencing the Van Hook reference, and appellant's arguments appear to be directed to his belief that Van Hook can not be combined to reject the claims. However, appellant is reminded that the rejection is Sidwell in view of Sun and that the Van Hook reference forms no part of the rejection of claim 16.

Response to appellant's brief section entitled: "1. Claims 21-22, 33-34 and 43-44 Are Not Obvious"

At pg. 44 of the brief, appellant argues:

"Sidwell does not disclose or suggest reuse of the multiply-add unit 76 even for performing packed multiply instructions (mul2ps)."

This is not found persuasive because appellant has also not claimed reuse of a multiply-add unit.

Appellant has merely claimed use of a multiplier, but not reuse of any multiplier.

"The invention disclosed in Hiniker's written description may be outstanding in its field, but the name of the game is the claim." *In re Hiniker Co.*, 47 USPQ2d 1523, 1529 (Fed. Cir. 1998)

At pg. 45 of the brief, appellant argues:

"Sun does not disclose or suggest a version of the vis_pdist() instruction that does not use an accumulation of prior absolute differences."

This argument has been treated in depth starting at pg. 23 and pg. 31 above.

At pg. 45 of the brief, appellant argues:

"Lee does not disclose or suggest use of partial product selectors to insert the elements of a packed data into bit positions of the partial products to add the elements together."

This argument is not found persuasive. As detailed starting at pg. 16 above in the rejection of claims 21, 33, and 43, Lee does show insertion of elements of packed data into bit positions of partial products to add the elements together. Note in table 5 at col. 5 of Lee, packed data "abcd" is inserted into partial product 1 (the second table row) while packed data "efgh" is inserted into partial product 5 (the sixth row), so that they can be summed together to produce sum "ZZZZZ" in the final row of the table.

At pg. 46 of the brief, appellant argues:

"Neither Sidwell nor Sun disclose a plurality of partial product selectors to insert elements of a packed data into and substituting for bit positions of one or more partial products and adding the elements together."

This argument is not persuasive because as detailed above starting on pg. 26, appellant is merely arguing the references individually, which is improper. Neither Sidwell nor Sun was relied upon in the rejection to show this insertion aspect.

At pg. 46 of the brief, appellant argues:

"Lee's method, on the other hand, generates control inputs to force to logic zero bit positions that do not correspond to the bit positions of an element to be added, rather than inserting elements of a packed data into and substituting for bit positions to be added as set forth in Claim 21."

This is not found persuasive as is clearly seen from tables 5 and 6 on cols. 5 and 6 of the Lee reference, Lee does indeed "insert elements of a packed data into and substituting for bit positions to be added" as claimed. Note that in table 5, in row two, the element "abcd" has been inserted into bit positions

eight through five (numbering from the right) while element "efgh" has been inserted into bit positions four through one. Table 6 shows "insertion" for four data elements instead of two. Lee then adds the bit positions into a sum "ZZZZZ" in table 5 and "ZZZZZZ" in table 6.

At pg. 46 of the brief, appellant argues:

"Further, Lee aligns data from one input in partial products through use of another input value rather than using partial product selectors corresponding to the bit positions to be added as set forth in Claim 21."

This is not found persuasive because as seen from table 5 and table 6 of Lee, and as detailed in the rejections above, Lee includes partial product selectors (column of bits running down the right hand side of the tables) that indicate where the insertion is to take place. Note that in each row where an element has been inserted, the selector is set to one, while in rows where no element is inserted, the selector is a zero. Accordingly, the one bits in the right most column are the partial product selectors that select (or "choose") where to insert the elements within the partial products.

At pg. 47 of the brief, appellant argues:

"The vis_pdist() instruction of Sun already has three source operands ... To perform the alignment in partial products as suggested by Lee a fourth source operand would be necessary. Sidwell's system provides no third path for source inputs to packed arithmetic unit 6, much less a fourth"

This is not found persuasive for two reasons. First appellant has merely put forth the mere unsupported argument that a fourth source operand would be necessary. As detailed in MPEP 2145, attorney argument can not take the place of factual evidence to rebut a prima facie case of obviousness:

2145 [R-2] Consideration of Applicant's Rebuttal Arguments

I. ARGUMENT DOES NOT REPLACE EVIDENCE WHERE EVIDENCE IS NECESSARY

Attorney argument is not evidence unless it is an admission, in which case, an examiner may use the admission in making a rejection. See MPEP § 2129 and § 2144.03 for a discussion of admissions as prior art.

The arguments of counsel cannot take the place of evidence in the record. In re Schulze, 346 F.2d 600, 602, 145 USPQ 716, 718 (CCPA 1965); In re Geisler, 116 F.3d 1465, 43 USPQ2d 1362 (Fed. Cir. 1997) ("An assertion of what seems to follow from common experience is just attorney argument and not the kind of factual evidence that is required to rebut a prima facie case of obviousness.").

Second, applicant's argument is that it would not be possible to bodily incorporate Lee's teachings into Sidwell, which as pointed out starting on pg. 25 above, is also improper.

At pg. 48 of the brief, appellant argues:

"Because Lee generates control inputs to force bit positions that do not correspond to the bit positions of an element to be added, forty-eight (48) bit positions are forced to logic zero in order to sum four 4-bit numbers (Table 6; cols. 6, lines 9-61). Thus circuitry at forth-eight (48) bit positions must be modified rather than circuitry at the sixteen (16) bit positions of the four 4-bit numbers being added. Such considerations are of practical significance."

This is not found persuasive because appellant is again arguing features of his invention which are unstated in his claim language, which as shown starting on pg. 21 above, is improper. Appellant's claims merely require "inserting" of an "element" of a plurality of elements "into and substituting for" bits of "partial products". As seen from table 5 of Lee at col. 5, "partial product" five (sixth row of table) has had "inserted" and "substituting" for bit positions four through one the "element" "efgh" from packed data "abcdefgh" shown in row one of the table. Accordingly, to the extent appellant has claimed his invention, Lee is disclosing the same system. Appellant's claims provide no limitation, and therefore do not limit, the invention to any particular manner of "insertion" and of "substitution" of the element. Any manner of "insertion" and any manner of "substitution" reads upon and is within the boundaries set out by appellant's broad claim language.

Appellant at pgs. 48-50 continues with the same argument presented above, and this argument is not persuasive for the same reason presented above, namely that the claim language does not limit the manner of "insertion" or "substitution" to appellants particular method disclosed in the specification.

Response to appellants section entitled: "2. Claims 23-24 Are Not Obvious"

At pg. 51 of the brief, appellant argues:

"Sidwell does not disclose or suggest producing a packed result having two distinct sums of products as a result of the multiply-add instruction."

This is not found persuasive because as detailed starting on pg. 17 above in the rejection of claim 23, Sidwell did indeed disclose a system and structure that performed the operations claimed. Sidwell produces a packed result having two distinct sums of products as the outputs of adders 118 and 116 on fig. 6. Sidwell continues further, however as pointed out starting on pg. 23 above, the transitional phrase "comprising" in appellant's claims allows Sidwell's system to include any amount of additional steps or structure while still reading upon the claim language.

At pg. 51 of the brief, appellant argues:

"Sun does not disclose or suggest a version of the vis_pdist() instruction that does not use an accumulation of prior absolute differences."

This argument has been treated in depth starting on pgs. 23 and 31 above.

At pg. 52 of the brief, appellant argues:

"Nor does Sun disclose or suggest a plurality of partial product selectors to insert an element of a plurality of elements of a packed data into and substituting for bit positions of one or more partial products and add the plurality of elements together."

This argument is not persuasive because appellant is arguing the references individually, which as detailed starting at pg. 26 above, is improper. The rejection of claim 23 is Sidwell in view of Sun and Lee, and Lee provides the above argued elements to the combination.

At pg. 52 of the brief, appellant argues:

"Lee does not disclose or suggest use of partial product selectors to insert the elements of a packed data into bit positions of the partial products to add the elements together."

This argument has been treated extensively starting at pg. 36 above.

At pg. 53 of the brief, appellant argues:

"Lee's method, on the other hand, generates control inputs to force to logic zero bit positions that do not correspond to the bit positions of an element to be added, rather than inserting elements of a packed data into and substituting for bit positions to be added as set forth in claim 21 [sic] (col. 5, lines 3-5). Further, Lee aligns data from one input in partial products through use of another input value rather than using the partial product selectors corresponding to the bit positions to be added as set forth in Claim 21 [sic] (col. 1 lines 47-55)."

This argument has been extensively treated starting at pg. 36 above.

At pg. 54 of the brief, appellant argues:

"The multiply-add of Sidwell does not add a first and second set of partial products and a third and fourth set of partial products produce [sic] a packed result having two distinct elements."

This is not found persuasive because as shown above starting at pg. 17 in the rejection of claim 23, Sidwell does indeed teach the steps claimed by appellant's claim language. As explained above, fig. 6 shows production of first through fourth partial products (outputs of multipliers 108 through 114) and shows adding together first and second partial products (adder 118) and adding together third and fourth partial products (adder 116) to produce a packed result (output of adders 118 and 116). Sidwell continues further, but as detailed starting at pg. 23, appellant's transitional phrase "comprising" allows Sidwell to continue further while still reading upon the claim language.

At pg. 55 of the brief, appellant argues:

"Thus, Sun teaches away from computing the sum of two products as set forth in the packed multiply-add instruction of Claim 23"

This is not found persuasive because as detailed above, Sidwell taught this portion of claim 23.

Therefore, Sun can not be found to teach away when the reference explicitly taught the claimed limitations.

(11) Related Proceeding(s) Appendix

Appellant's brief contains a related proceedings appendix that states that appellant is not aware of any related appeals, interferences, or Judicial proceedings that would have a bearing upon the board's decision.

For the above reasons, it is believed that the rejections should be sustained.

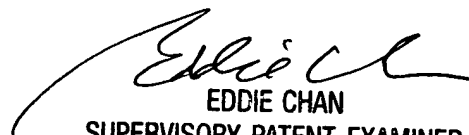
Respectfully submitted,

Richard Ellis

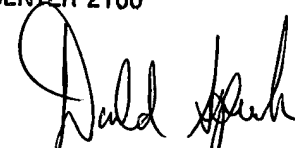

RICHARD L. ELLIS
PRIMARY EXAMINER

Conferees:

Eddie Chan
SPE AU 2183


EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Donald Sparks
SPE AU 2183


DONALD SPARKS
SUPERVISORY PATENT EXAMINER

Serial Number 10/005,728
Art Unit 2183
Paper Number

42 of 42

Evidence Appendix

Lee, Ruby B. (Lee(2)), *Subword Parallelism with MAX-2*, IEEE, August 1996.
Cited as evidence of the level of knowledge of one of skill in the art.